IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Robert T. George, et al.	§	Group Art Unit:	2181
		§		
Serial No.:	10/630,287	§		
		§	Examiner:	Niketa I. Patel
Filed:	July 30, 2003	§		
		§		
For:	Dynamically Partitioning Pipeline Resources	§	Atty. Dkt. No.:	ITL.1004US (P16592)
		§		

Mail Stop **Appeal Brief-Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Date of Deposit: October 25, 2006
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

Stephanie Petreas

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation, the assignee of the present application by virtue of the assignment recorded at Reel/Frame 014354/0955.

II. RELATED APPEALS AND INTERFERENCES

This appeal may relate to the co-pending appeal in U.S. Patent Application No. 10/630,286 filed July 30, 2003 entitled "Associating Address Space Identifiers With Active Contexts" in the names of Robert T. George, Jason W. Brandt, Jonathan D. Combs, Peter J. Ruscito, and Sanjoy K. Mondal.

III. STATUS OF CLAIMS

Claims 1-13 and 17-30 stand rejected. Claims 14-16 have been cancelled. The rejections of pending claims 1-10 and 17-30 are being appealed.

IV. STATUS OF AMENDMENTS

Appellants filed a "Reply to Final Office Action Mailed May 31, 2006" on July 25, 2006 including an amendment that cancelled claims 11-13. In the Advisory Action dated August 11, 2006, the Examiner did not indicate whether the amendment of that Reply was entered or not, however, according to private PAIR it was not entered.

Filed herewith is an Amendment Filed With Appeal Brief Under 37 C.F.R. §41.33, which cancels claims 11-13.

V. SUMMARY OF CLAIMED SUBJECT MATTER

At this point, no issue has been raised that would suggest that the words in the claims have any meaning other than their ordinary meanings. Nothing in this section should be taken as an indication that any claim term has a meaning other than its ordinary meaning.

Independent claim 1 is an independent method claim. Support for the subject matter of independent claim 1 can be found, for example, in the Specification as filed at p. 13, ln. 14 – p. 15, ln. 3.

Independent claim 6 is an independent method claim. Support for the subject matter of claim 6 can be found, for example, in the Specification at p. 5, lns. 11-18 and p. 12, ln. 1 – p. 14, ln. 3.

Independent claim 17 is an independent apparatus claim. Support for the subject matter of independent claim 17 can be found, for example, in the Specification at p. 9, ln. 4 – p. 11, ln. 6.

Independent claim 21 is an independent method claim. Support for independent claim 21 can be found, for example, in the Specification at p. 3, ln. 17 – p. 4, ln. 2.

Independent claim 25 is an independent article claim. Support for independent claim 25 can be found in the Specification as filed, as example, at p. 3, ln. 17 – p. 4, ln. 2, along with p. 23, ln. 13 – p. 24, ln. 2.

Independent claim 28 is an independent system claim. Support for independent claim 28 can be found, for example, in the Specification as filed at p. 9, ln. 4 – p. 11, ln. 6. Furthermore, additional support for claim 28 may be found at FIG. 5 and the corresponding portion of the Specification, namely p. 24, ln. 3 – p. 26, ln. 3.

The following is a further concise explanation of the subject matter defined in the independent claims of the present appeal, also identified by page and line number in the specification.

In various embodiments of the present invention, pipeline resources or structures may be dynamically partitioned to provide support for multiple address spaces. By "dynamically," it is meant that the partitioning of pipeline resources may be modified during operation such that at different times, different address spaces may occupy a given partition of a pipeline resource and/or that different partitions may consume more or less of the resource. As used herein, the

term "address space" means a set of addresses in memory corresponding to a given application (e.g., a context). Specification, p. 3, ln.17 – p. 4, ln. 2.

Various embodiments may also include a store filter (also termed a "flush filter" herein) which monitors stores for updates to page tables, and may selectively invalidate associated translations. address space identifiers (ASIDs) or address space numbers (ASNs) may be used to augment linear addresses in various pipeline resources with a pointer to the context with which they are associated. An "address space identifier" or "address space number" may be any number, code, or other notation which identifies one or more address spaces with which it is associated. This allows multiple application contexts to share pipeline structures, reducing context-switch overhead. For example, when a context switch occurs, the current ASID value is changed, rather than flushing the pipeline structures. Similarly, in certain embodiments, a thread identifier (thread ID) may be provided to identify a given processor thread for a corresponding address space. Specification, p. 4, ln. 14 – p. 5, ln. 10.

In certain embodiments, the flush filter may monitor updates to intermediate page table entries, and selectively flush TLB entries corresponding to a specific ASID. Various architectural events may cause a selective flush of the TLBs in accordance with embodiments of the present invention. Specification, p. 5, lns. 11-18.

In various embodiments, ASIDs may be used to augment the linear address in pipeline resources with a pointer to the corresponding address space. In such embodiments, the microprocessor may maintain a global current ASID register (or ASID manager) that is updated when a new address space is created or when changing to a different, previously seen address space. TLB insertions may be extended with the current ASID value, and TLB lookups match only if the ASID tag matches the current ASID value. When a context switch triggers an address space change, the microprocessor may switch to a different ASID value that represents the new address space, instead of flushing the TLB's and other pipeline structures. In certain embodiments, selectively flushing entries corresponding to a specific address space may provide a substantial performance gain for environments with multiple contexts. Specification, p. 6, lns. 2-18.

In one embodiment, a flush filter may be implemented as a content addressable memory (CAM) which matches post-retirement store addresses (i.e., physical addresses) against known page directory base and page table bases to determine if the store targets a page table belonging

to one of the ASID contexts. If the flush filter finds a match, the store is attempting to update a page table entry cached in the TLB, and the flush filter provides the corresponding ASID value and thread. The associated ASID partition may then be marked as hit/modified, and the partition flush may be delayed until the next TLB flushing event. As a result, the flush filter identifies all page table updates that would change a currently cached TLB entry, and entries corresponding to a specific address space are invalidated. The resulting ASID flush (essentially a selective TLB flush) may be delayed until a context switch (i.e., a next TLB flushing event). Specification, p. 6, ln. 19 – p. 7, ln. 10.

In various embodiments, during steady state operation (i.e., after the flush filter has already been filled), the flush filter may monitor updates to page tables by filtering addresses of senior stores and external snoops. The physical address bits of these operations may be CAM'ed against the base address stored in the flush filter. The flush filter may be filled during TLB page walks. On every page walk, addresses representing page directory bases and page table bases, along with the current thread ID and ASID, may be sent to the flush filter. For each address sent to the flush filter during the fill, it may check if that address is already stored for the same (current) Thread and ASID context. If there is not an exact match (including the case where the physical address matches but the thread ID or ASID do not), then the flush filter allocates a new entry in the CAM with the address as the tag, and the thread ID and ASID as the match data, regardless of whether a global bit is set, meaning the page translation is used in another context. Specification, p. 8, ln. 4 – p. 9, ln. 3.

Referring now to FIG. 1B, shown is a block diagram of a flush filter in accordance with one embodiment of the present invention. As shown in FIG. 1B, a portion of a processor pipeline includes a DAC multiplexer 15, a filter 20, a store buffer 30, a flush vector control register (or "flush vector") 35, a page miss handler (PMH) 40, and a second multiplexer 45. Flush vector 35 may receive hits of a CAM match (i.e., tag address, ASID, and thread ID) via line 22 from filter 20. As shown in the embodiment of FIG. 1B, each entry of filter 20 may include a first valid bit (V0), a tag address. Additionally, the entry may include a second valid bit (V1). In certain embodiments, the presence of two valid bit arrays may be used to handle filter fill operations while a particular partition of the filter is being invalidated. Entries may further include an ASID and a thread ID, which are shown as N-bit and M-bit numbers,

respectively. As shown in FIG. 1B, tag addresses may be stored in filter 20 via line 25 which is received in a fill port of filter 20. Specification, p. 9, ln.4 – p. 10, ln. 18.

As further shown in FIG. 1B, a multiplexer 45 may be coupled to filter 20 via line 28 to provide signals to an invalidate port of filter 20. In one embodiment, microcode instructions may be used to invalidate entries of filter 20. Specification, p. 11, lns.1-5.

Referring now to FIG. 2, shown is a block diagram of flush filter system in accordance with one embodiment of the present invention. As shown in FIG. 2, a flush filter 20 may receive tag addresses from page miss handler 40. For each entry filled into filter 20, an ASID manager 45 may provide a current ASID for the appropriate address space. ASID manager 45 may store various CR3 values and provide an associated ASID for a present context.

When flush filter 20 compares senior store addresses or external snoops to tag addresses stored therein and a CAM match is found, a hit for the matched entry in flush vector 35 may be provided to flush filter controller 60. Flush filter controller 60 may be used to provide invalidate instructions to flush filter 20, and to send flush instructions to associated TLBs. As shown in FIG. 2, these TLBs may include an iTLB 50, a dTLB 70 and a STLB 75. Of course, these TLBs may have extended entries that include ASIDs and thread IDs. Specification, p. 11, lns. 1-26.

Referring now to FIG. 3, shown is a flow diagram of a method in accordance with one embodiment of the present invention. As shown in FIG. 3, the method begins by obtaining a physical address (block 110). Next, the filter may compare the physical address obtained to tag addresses in the filter (block 120). Such comparison may be performed to determine whether there is a match between the physical address and any of the tag addresses stored in the filter and corresponding ASID and thread ID (diamond 130). If no such matches occur, control returns to block 110 where another physical address may be obtained. If one or more matches is found, valid bit V0 may be set if there is no conflicting invalidating access occurring, and valid bit V1 may be set if there is a conflicting invalidating access occurring. Specification, p. 12, ln. 1 – p. 13, ln. 6.

If there is a match, a control register coupled to the filter may be updated (block 140). Still referring to FIG. 3, the address space or spaces of the filter corresponding to the updated address spaces in flush vector 35 may be invalidated (block 150). In certain embodiments, microcode may read the contents of flush vector 35 and determine that a particular ASID and thread ID have been set and perform an invalidation process. In such a process, microcode may

cause the V0 array to be invalidated at a cycle n and the V1 array to be invalidated at a cycle n+1. Further, microcode may cause flush vector 35 to be cleared. Thus for a given ASID and thread ID, when an update is detected by filter 20, entries in filter 20 corresponding to the particular ASID and thread ID may be invalidated. Finally, on the next context switch the invalidated address space may be flushed from the filter and associated pipeline structures, such as the TLB (block 160). Specification, p. 13, ln. 7 – p. 14, ln. 3.

Referring now to FIG. 4, shown is a block diagram showing correspondence between a flush filter and a TLB in accordance with one embodiment. As shown in FIG. 4, flush filter 20 includes physical addresses (tag addresses) and associated valid, thread ID and ASID bits. Filter 20 may be coupled to an associated TLB 50 which is similarly augmented to include the valid bit, thread ID, and ASID, along with corresponding linear and physical addresses. Specification, p. 16, lns. 9-17.

Since there are a large number of possible base combinations for a single address space, in certain embodiments the flush filter may be partitioned to prevent a greedy application from consuming all the flush filter match slots. Two flush filter partitioning mechanisms may be applied to prevent ASID capacity issues: static partitioning and dynamic partitioning, both of which may provide for overflow conditions. In certain embodiments, the flush filter may be segmented with a static partition per ASID. In an embodiment having a 32-entry CAM, a hard-partitioned flush filter with a 2-bit ASID per thread ID would allow 4 CR3/PD base/PT base entries per ASID/thread ID. In an embodiment having a 32-entry CAM with dynamic partitioning, a 2-bit ASID per thread ID would allow any ASID/thread ID context to consume a variable number of CR3/PD base/PT base entries. Specification, p. 19, lns. 1-17.

Referring now to FIG. 5, shown is a block diagram of a representative multiprocessor computer system 200 in accordance with one embodiment of the invention. As shown in FIG. 5, multiprocessor computer system 200 includes a first processor 201 and a second processor 211. Processors 201 and 211 may be coupled over a front-side bus 220 to a memory hub 230 in one embodiment, which may be coupled to a shared main memory 240 via a memory bus. Memory hub 230 may also be coupled (via a hub link) to an input/output (I/O) hub 235 that is coupled to an I/O expansion bus 255 and a peripheral bus 250. In various embodiments, I/O expansion bus 255 may be coupled to various I/O devices such as a keyboard and mouse, among other devices. Peripheral bus 250 may be coupled to various components such as peripheral device 270 which

may be a memory device such as a flash memory, add-in card, and the like. Specification, p. 24, lns. 3-19.

As shown in FIG. 5, first processor 201 may include a TLB 203 and a filter 205 in accordance with an embodiment of the present invention. More so, a level 2 (L2) cache 207 may be coupled to processor 201. Similarly, second processor 211 may include a TLB 213, a filter 215 and may be coupled to a L2 cache 217. Specification, p. 24, ln. 24 – p. 25, ln. 3.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Each of the following grounds of rejection are presented for review:

(1) claims 1-3 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(2) claim 4 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(3) claim 5 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(4) claims 6 and 8-10 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(5) claim 7 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(6) claims 17-20, and 28-30 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(7) claims 21-22 and 25-26 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(8) claim 23 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(9) claims 24 and 27 stand rejected under 35 U.S.C. §102(e) over Zuraski;

VII. ARGUMENT

(1) Claims 1-3 Are Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 1 is an independent method claim that recites invalidating an entry of a filter coupled to a pipeline resource if an update to the entry occurs during a context, and flushing a portion of the pipeline resource corresponding to an address space that includes the entry. Pending claim 1 stands rejected under 35 U.S.C. §102(e) over U.S. Patent No. 6,510,508 (Zuraski). The rejection is clearly erroneous, as Zuraski fails to teach each and every element of the claims, as required for a valid anticipation rejection.

As to claim 1, Zuraski nowhere teaches either: (1) invalidating an entry of a filter if an update to the entry occurs during a context; or (2) flushing a portion of a pipeline resource coupled to the filter corresponding to an address space including the entry.

In this regard, the Examiner refers to col. 9, lns. 48-61 and col. 11, lns. 14-20 of Zuraski to contend presence of the claimed invalidating of an entry of a filter. The Examiner contends that flush filter 40 is the recited filter. While this may be, there is simply no teaching that an entry in this filter is invalidated if an update occurs to the entry. That is, nowhere do these cited portions or any other portion of Zuraski anywhere teach that an entry in flush filter 40 is invalidated if an update to the entry occurs during a context. Instead, both of these portions only teach that a flush filter 40 asserts an Invalidate signal out to a translation lookaside buffer (TLB) 39. Nowhere do these or any other portions of Zuraski teach that such Invalidate signal invalidates an entry of the filter itself. Certainly, Zuraski nowhere teaches invalidating such an entry if an update to the entry occurs during a context.

Furthermore, nowhere does Zuraski anywhere teach flushing *a portion of* a pipeline resource corresponding to an address space including such an entry. In this regard, the Examiner refers to col. 13, lns. 3-11 of Zuraski. However, neither this nor any other portion of Zuraski teaches such flushing of a portion of a resource. Instead, all that Zuraski teaches is that upon the occurrence of certain events, flush filter 40 causes a flush of TLB 39 *in its entirety*. That is, Zuraski instead teaches:

> In the embodiment shown, a flush of TLB may occur when filter circuit 403 asserts an Invalidate signal, *thereby invalidating all entries currently stored in TLB 39.*

Zuraski, col. 11, lns. 14-17 (emphasis added). Clearly, Zuraski teaches that the *entire* TLB is flushed.

Nor does the passage of Zuraski cited in col. 13 anywhere teach flushing of *a portion of* a pipeline resource. Instead, this portion of Zuraski merely teaches:

> As such, it is possible that some address translations loaded into the TLB from page table B are no longer valid. Consequently, *the TLB flush filter may allow a TLB flush* (shown here as a filtered flush) to occur following the next context switch.

Zuraski, col. 13, lns. 7-11 (emphasis added). Clearly, this does not teach the flushing of a portion; instead it is a full TLB flush.

For at least these reasons, Zuraski does not teach flushing of only a portion of a pipeline resource, and certainly not a portion corresponding to an address space including an invalidated and updated entry of a filter. Accordingly, the rejection of claim 1 and its dependent claims is improper and the rejection should be reversed.

(2) Claim 4 Is Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 4 depends from claim 1 and further recites comparing an address obtained from an external snoop to multiple entries in the filter to determine if an update to an entry in the filter has occurred. Claim 4 also stands rejected under 35 U.S.C. § 102(e) over Zuraski. The rejection is improper at least for the same reasons discussed above regarding claim 1 (*see* VII.1.).

Dependent claim 4 is further patentable, as Zuraski nowhere teaches comparing an address obtained from an external snoop to entries in the filter to determine if an update has occurred. In this regard, the Examiner refers to col. 10, lns. 1-17. All this portion of Zuraski teaches is that a flush filter 40 may receive a snoop request signal and a snoop address. However, Zuraski only teaches that this information is used "for searches of a region table." Zuraski does not teach that an entry in a flush filter is invalidated based on a comparison of an address obtained from an external snoop, and thus the rejection should be reversed.

(3) Claim 5 Is Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 5 depends from claim 1 and further recites that the portion of the pipeline resource is flushed via microcode. Claim 5 also stands rejected under 35 U.S.C. §102(e) over Zuraski.

This rejection is improper at least for the same reasons discussed above regarding claim 1 (*see* VII.1.).

The rejection is further improper, as Zuraski does not teach that its flush operation is performed by microcode, contrary to the Examiner's contention. Instead, Zuraski merely teaches that an Invalidate signal is asserted to allow a flush of TLB 39. Zuraski, col. 9, lns. 53-55. To support the rejection of claim 5, the Examiner instead refers to an unrelated portion of Zuraski: col. 4, lns. 11-29. However, all this teaches is that an instruction cache includes tags as to whether an instruction is executed by invoking a microcode procedure. This portion has no relevance to the operation of the flush filter, as there is no indication that the Invalidate signal is generated in any way by execution of an instruction. As a result, there is no teaching of the claim 5 subject matter as recited in the claim, and accordingly claim 5 is patentable for this further reason. *In re Bond*, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990) (every element of claim must be identically shown in a single reference).

(4) Claims 6 and 8-10 Are Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 6 is an independent method claim that recites flushing a portion of entries of a pipeline resource if one of the portion of entries is updated during a context. Claim 6 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper as Zuraski nowhere teaches flushing a portion of entries of a pipeline resource if one of the portion is updated during a context.

For claim 6, the Examiner simply refers to the same portions of Zuraski cited above with regard to claim 1. Zuraski, however, does not teach that its TLB has a portion that is flushed if one of its entries is updated during a context. Instead, the entire TLB 39 of Zuraski is flushed if any entry in the TLB is updated during a context. That is, as described above with regard to claim 1 (*see* VII.1.), Zuraski teaches that flush filter 40 causes a flush *of the entire contents* of TLB 39. Zuraski, col. 11, lns. 14-17. Thus, the rejection of claim 6 and its dependent claims is clearly erroneous, and the rejection should be reversed.

(5) Claim 7 Is Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 7 depends from claim 6 and further recites selectively flushing the portion of entries of the pipeline resource on a switch from the context. Dependent claim 7 stands rejected

under 35 U.S.C. §102(e) over Zuraski. This rejection is improper at least for the same reasons discussed above regarding claim 6 (*see* VII.4.).

Furthermore, Zuraski nowhere teaches selective flushing. Instead, all that Zuraski teaches is that an Invalidate signal is sent from flush filter 40 to TLB 39, which causes a complete flush of the TLB: "a flush of TLB may occur when filter circuit 403 asserts an Invalidate signal, thereby *invalidating all entries* currently stored in TLB 39." Zuraski, col. 11, lns. 14-17 (emphasis added). Thus there is no selective flushing in Zuraski and claim 7 is patentable for this further reason.

(6) Claims 17-20 and 28-30 Are Patentable Under 35 U.S.C. § 102(e) over Zuraski

Independent claim 17 recites an apparatus that includes a pipeline resource having multiple address spaces, each corresponding to one of multiple contexts, where each of the address spaces is selectively flushable while the other address spaces are maintained. Claim 17 stand rejected under 35 U.S.C. § 102(e) over Zuraski. This rejection is improper, as Zuraski nowhere teaches a pipeline resource having multiple address spaces that are each selectively flushable while the others are maintained. Instead, as described above with respect to claim 1 (*see* VII.1.), the TLB of Zuraski is flushed in its entirety.

To support the rejection, the Examiner merely relies on col. 13, lns. 3-11 to contend such maintaining of entries. As discussed above, this portion of Zuraski, as with the rest of Zuraski, teaches an entire flush of the TLB 39. Accordingly, the rejection of claim 17 and its dependent claims is clearly erroneous and the rejection should be reversed. For at least the same reasons, the rejection of independent claim 28 and its dependent claims should also be reversed.

(7) Claims 21-22 and 25-26 Are Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 21 is an independent method claim that recites dynamically partitioning a filter of a pipeline resource into multiple partitions, where each of the partitions corresponds to one of multiple address spaces. Claim 21 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper, as Zuraski nowhere teaches either dynamic partitioning of a filter into multiple partitions, nor where such partitions each correspond to one of multiple address spaces.

In this regard, there is no teaching anywhere in Zuraski of dynamic partitioning. Instead, the cited portion of Zuraski (i.e., col. 10, lns. 27-65) merely sets forth the fixed structure of the

flush filter of Zuraski. That is, while the circuitry of the flush filter shown in FIG. 3 of Zuraski and described in col. 10 includes different components, Zuraski nowhere teaches that these components are dynamically partitioned. Instead, it appears the opposite is true: the structures detailed in col. 10 of Zuraski are fixed and thus are not dynamically partitioned.

Further, Zuraski does not teach that these components are partitioned into partitions each corresponding to one of multiple address spaces. Instead, all that the Examiner cites in this regard is that a region table includes a CAM and a RAM to store addresses and tags. Nowhere however does Zuraski teach that such storage is partitioned so that each partition corresponds to only one of multiple address spaces. Accordingly, the rejection of claim 21 and its dependent claims is clearly erroneous. For at least similar reasons, so too is the rejection of claim 25 and its dependent claims clearly erroneous.

(8) Claim 23 Is Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 23 depends from claim 22 which in turn depends from claim 21. Claim 23 further recites that each of the multiple partitions of the filter includes a fixed portion, and that the filter further includes a dynamic portion. Claim 23 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper at least for the same reasons discussed above regarding claim 21 (*see* VII.7.).

As contended support for this recited subject matter, the Examiner refers to a paragraph in the Summary of Invention (specifically lns. 36-51 of col. 2). Nowhere does this or any other portion of Zuraski anywhere teach that the flush filter 40 includes fixed partitions and a dynamic portion. Accordingly, for this further reason the rejection of claim 23 is clearly erroneous and the rejection should be reversed.

(9) Claims 24 and 27 Are Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 24 depends from claim 23 and further recites allocating at least part of the dynamic portion of the filter to an application that has consumed the fixed portion of its partition. Claim 24 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper at least for the same reasons discussed above regarding claim 23 (*see* VII.8.).

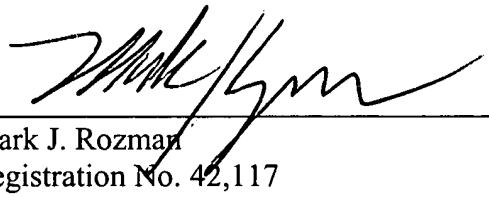
The rejection is further improper as Zuraski further nowhere teaches allocating at least part of the dynamic portion to an application that has consumed the fixed portion of its partition.

In fact, Zuraski appears to teach the opposite. That is, Zuraski teaches that when the region table of the flush filter 40 runs out of available resources an overflow signal is asserted, and that this overflow signal deactivates the flush filter 40. Zuraski, col. 10, lns. 49-65.

Clearly, there is no teaching of allocating a dynamic portion if an application has consumed a fixed portion. Instead, an overflow occurs and the flush filter of Zuraski is simply deactivated. For this further reason the rejection of claims 24 and 27 is clearly erroneous and should be reversed.

Appellants respectfully request that each of the final rejections be reversed and that the claims subject to this Appeal be allowed to issue.

Respectfully submitted,



Date: October 25, 2006

Mark J. Rozman
Registration No. 42,117
TROP, PRUNER & HU, P.C.
1616 S. Voss Road, Suite 750
Houston, Texas 77057-2631
(512) 418-9944 [Phone]
(713) 468-8883 [Fax]
Customer No.: 21906

VII. CLAIMS APPENDIX

The claims on appeal are:

1. A method comprising:

invalidating an entry of a filter coupled to a pipeline resource if an update to the entry occurs during a context; and

flushing a portion of the pipeline resource corresponding to an address space including the entry.

2. The method of claim 1, further comprising flushing the portion upon a switch from the context.

3. The method of claim 1, wherein the pipeline resource comprises a translation lookaside buffer.

4. The method of claim 1, further comprising comparing an address obtained from an external snoop to a plurality of entries in the filter to determine if the update has occurred.

5. The method of claim 1, further comprising flushing the portion of the pipeline resource via microcode.

6. A method comprising:

flushing a portion of entries of a pipeline resource if one of the portion of entries is updated during a context.

7. The method of claim 6, further comprising selectively flushing the portion of entries upon a switch from the context.

8. The method of claim 7, wherein the portion of entries comprises an address space corresponding to the context.

9. The method of claim 6, wherein the pipeline resource comprises a translation lookaside buffer.

10. The method of claim 9, further comprising invalidating entries of a filter coupled to the translation lookaside buffer corresponding to the portion of entries.

11. A method comprising:

loading an entry into a pipeline resource of a processor, the entry corresponding to a page table; and

selectively flushing the entry if the page table is updated during a context.

12. The method of claim 11, further comprising selectively flushing the entry upon a switch from the context.

13. The method of claim 11, further comprising invalidating a filter entry of a filter coupled to the pipeline resource, the filter entry corresponding to the entry.

17. An apparatus comprising:

a pipeline resource having a plurality of address spaces, each of the plurality of address spaces corresponding to one of a plurality of contexts, each one of the plurality of address spaces selectively flushable while the other address spaces are maintained.

18. The apparatus of claim 17, wherein the pipeline resource comprises a translation lookaside buffer.

19. The apparatus of claim 17, further comprising a filter coupled to the pipeline resource to select at least one of the plurality of address spaces to be flushed.

20. The apparatus of claim 19, wherein the filter comprises a content addressable memory.

21. A method comprising:

dynamically partitioning a filter of a pipeline resource into a plurality of partitions, each of the partitions corresponding to one of a plurality of address spaces.

22. The method of claim 21, further comprising sharing the pipeline resource among a plurality of applications, each corresponding to one of the plurality of address spaces.

23. The method of claim 22, wherein each of the plurality of partitions includes a fixed portion and wherein the filter further comprises a dynamic portion.

24. The method of claim 23, further comprising allocating at least part of the dynamic portion to one of the plurality of applications that has consumed the fixed portion of one of the plurality of partitions.

25. An article comprising a machine-readable storage medium containing instructions that if executed enable a system to:

dynamically partition a filter of a pipeline resource into a plurality of partitions, each of the partitions corresponding to one of a plurality of address spaces.

26. The article of claim 25, further comprising instructions that if executed enable the system to permit a plurality of applications, each corresponding to one of the plurality of address spaces, to share the pipeline resource.

27. The article of claim 26, further comprising instructions that if executed enable the system to allocate at least part of a dynamic portion of the filter to one of the plurality of applications that has consumed one of the plurality of partitions.

28. A system comprising:

a first processor having a pipeline resource having a plurality of address spaces, each of the plurality of address spaces corresponding to one of a plurality of contexts, each one of the plurality of address spaces selectively flushable while the other address spaces are maintained; and

a dynamic random access memory coupled to the first processor.

29. The system of claim 28, further comprising a second processor coupled to the first processor.

30. The system of claim 29, further comprising a filter coupled to the pipeline resource to snoop address information from the second processor.

IX. EVIDENCE APPENDIX

There was no evidence submitted during prosecution.

X. RELATED PROCEEDINGS APPENDIX

There is no decision rendered by a court or the Board in the Appeal identified in Section II, above.